UNIVERSITY OF TORONTO – CSC 373 – JULY 20, 2006

TEST 2

Student Number: _____

Email Address: _____

Last (Family) Name(s): _____

First (Given) Name(s): _____

Tutorial (circle): M. Demko (BA3012) M. Fazel-Zarandi (BA3116)

- This test is worth 15% of your final mark.
- Answer each question directly on the test paper, in the space provided. Use the reverse side of the pages for rough work. If you need more space for one of your solutions, use the reverse side of a page and *indicate clearly the part of your work that should be marked.*
- **20% rule:** If you are unable to answer (a part of) a question, you will get 20% of the marks for the (part of the) question if you write "I don't know" and *nothing else* for that part/question.

The Master Theorem

If for some $a, b, d > 0$:

$$T(n) = aT(n/b) + \mathcal{O}(n^d)$$

then

$$T(n) = \begin{cases} \mathcal{O}(n^d) & \text{if } a < b^d \\ \mathcal{O}(n^d \log(n)) & \text{if } a = b^d \\ \mathcal{O}(n^{\log_b(a)}) & \text{if } a > b^d \end{cases}$$

Question 1 (out of 14)	
Question 2 (out of 22)	
Total (out of 36)	

Question 1 [14]

Given a set P of n points on the two dimensional plane:

$$p_i = (x_i, y_i) \qquad \text{for } \; i = 1, \ldots, n,$$

where $x_i \neq x_j$ for $i \neq j$, the Closest Pair Problem is to find a pair p_i, p_j that has smallest distance. This question is about the following divide-and-conquer algorithm that solves this problem in time $\mathcal{O}(n \log(n))$. The algorithm works as follows: First, before calling the recursive divide-and-conquer procedure, it performs some preprocessing on the input P. Then it calls a recursive procedure which consists of the following steps:

1. Divide P into two halves Q, R by a vertical line ℓ.
2. Compute recursively the closest pairs for Q and R. Let δ be the distance between the closest of these two pairs.
3. If there are pairs (p_i, p_j) where $p_i \in Q$ and $p_j \in R$ that have distance less than δ, then find the closest of such pairs.
4. If step 3 returns a pair, then output that pair. Otherwise output the closest pair from step 2.

The preprocessing consists of sorting P into different order. Part **a)** and **b)** below are about this part of the algorithm. You do *not* have to give pseudo-code for the sorting procedures.

a [**2**] How can we preprocess the input P so that step 1 above can be done in time $\mathcal{O}(n)$?

b [**2**] How can we preprocess the input P so that step 3 above can be done in time $\mathcal{O}(n)$?

c [**5**] Explain why step 3 can be done in time $\mathcal{O}(n)$.

d [**5**] Give pseudo-code for the above algorithm. Include the preprocessing part as well as the recursive procedure. Again, you do *not* have to give pseudo-code for the sorting procedures.

Question 2 [22]

Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible *base stations*. There are n clients $C_1, C_2, \ldots, C_n$, with the position of each client specified by its (x, y) coordinates in the plane. There are also k base stations $B_1, B_2, \ldots, B_k$; the position of each of these is specified by its (x, y) coordinates as well.

For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways:

- There is a *range parameter* r: A client can only be connected to a base station that is within distance r.
- There is also a *load parameter* L: no more than L clients can be connected to any single base station.

The Client-Station problem is, given the positions of a set of clients and a set of base stations, to find a set of *maximum* number of clients that can be connected simultaneously to the stations.

In this question you are asked to design an algorithm for the above problem using flow network.

a [5] Construct a flow network that can be used to solve this problem. Precisely specify the nodes, the edges (with directions) and the capacities in the network.

b [2] Name an algorithm to find the maximum flow in the network. (You do *not* have to give pseudo-code for the algorithm.)

c [10] What is the relationship between the maximum flow and the maximum number of clients that can be connected simultaneously to the stations? Prove. [Clearly state the facts that you need about the output of the algorithm you gave in part **b)**. You are *not* required to prove these facts.]

(Question **2c** continues here)

d [**2**] Using the maximum flow returned in step 2, show how to compute *a set of maximum number of clients* that can be connected simultaneously to the stations.

e [**3**] What are the running time of the procedures in parts **a)**, **b)** and **d)** ? What is the running time of your algorithm–i.e., the total running time of the procedures in parts **a)**, **b)** and **d)** ? State your answers using Θ notation.